ARRAYLIST

01 March 2025 12:58

➢ **What is ArrayList?**

- ArrayList is a **resizable array** (dynamic array) in Java.
- It **automatically grows or shrinks** as we add/remove elements.
- It **comes from** the **Collection Framework**.
- It is **part of java.util package**.
- It **implements List interface**, so it keeps the **insertion order** (order of elements remains the same).

Why do we need ArrayList?

💡 **Problem with Array**

- Array has **fixed size** — once created, you **cannot increase or decrease its length**.

```
int[] arr = new int[5];  // Fixed size array
```

➢ **Solution - ArrayList**

- ArrayList handles size **automatically**.
- Add or remove any time — no manual resizing.

➢ **Key Features of ArrayList**

✅ Dynamic resizing
✅ Maintains insertion order
✅ Allows duplicates
✅ Allows null values
✅ Random access (index-based like array)

🔷 Real World Example - Where to Use?

Use Case	Example
✅ Shopping Cart	Add/remove items dynamically.
✅ Attendance List	Add student names, mark present/absent.
✅ Dynamic Menu	Add/remove menu items from a list.
✅ Form Selection	Store user-selected items in a list.

➢ Creating ArrayList (Syntax)

a.
```
ArrayList<String> list = new ArrayList<>();
```

🔶 **Advantages of ArrayList**

✅ Easy to use (like normal array)
✅ Dynamic resizing (no size headache)
✅ Lots of ready methods in Collection framework

✅ Supports all data types using Generics

🧩 **Disadvantages of ArrayList**

❌ Slow if adding/removing in **middle** (because shifting needed)

❌ Not thread-safe (needs manual synchronization for multi-threading)

❌ Slightly slower than array for **fixed-size data** (due to resizing overhead)

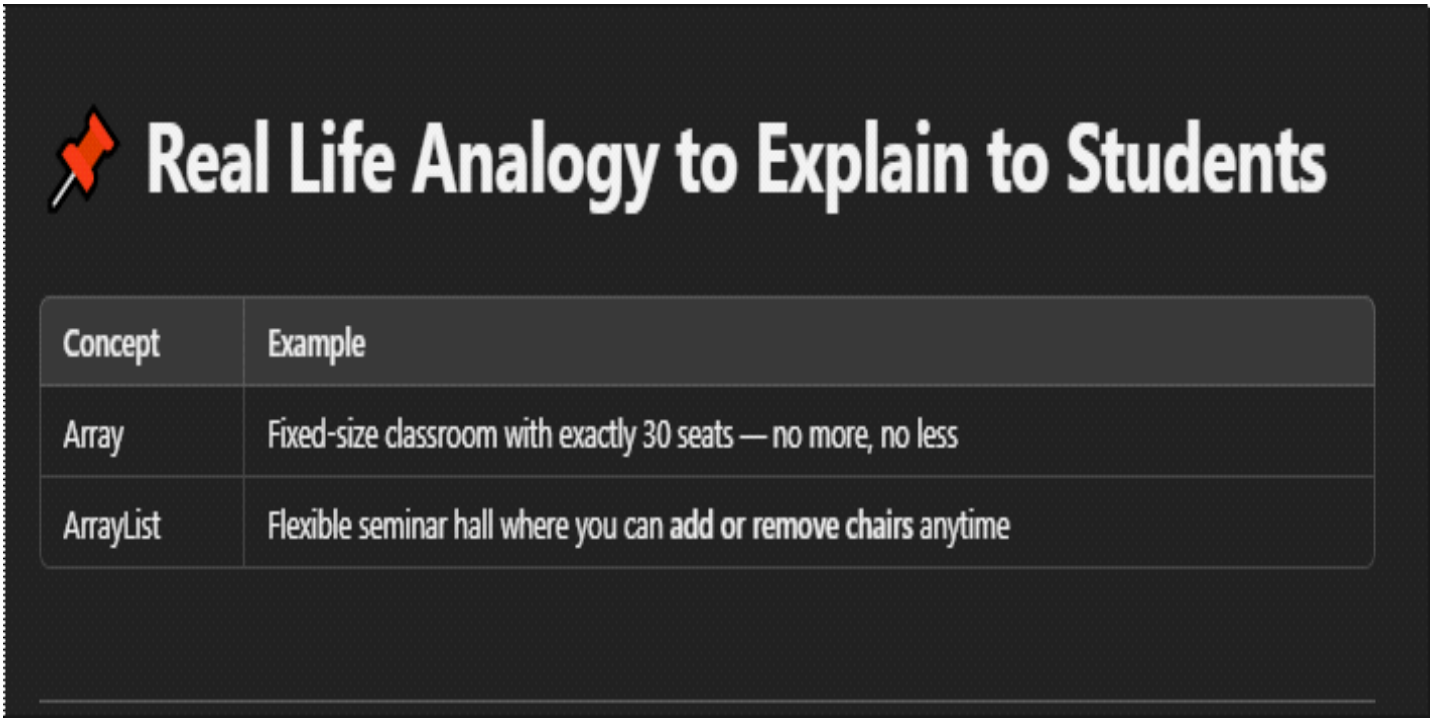
🔷 When to Use ArrayList?

Use Case	Recommended?
✅ Data is dynamic	Yes
✅ Order matters	Yes
✅ Duplicates allowed	Yes
❌ High-speed insert/remove at start/middle	No (Use `LinkedList`)
❌ Multi-threading	No (Use `Vector` or `Collections.synchronizedList()`)

x

🔷 Quick Difference: Array vs ArrayList

Feature	Array	ArrayList
Size	Fixed	Dynamic
Type	Primitive + Objects	Objects only
Performance	Faster (direct memory)	Slightly slower (resizing)
Flexibility	Less	More
Part of	Core Java	Collection Framework



CODE:

```
package collection_framework;
import java.util.ArrayList;

public class ArrayListDemo {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();

        list.add("Apple");
        list.add("Banana");
        list.add("Mango");

        System.out.println("Original List: " + list);

        // ArrayList does not have addFirst() or addLast()
        list.add(0, "Orange");   // Equivalent to addFirst()
        list.add("Grapes");      // Equivalent to addLast()

        System.out.println("After addFirst and addLast (manually): " + list);

        // Removing first and last element
        list.remove(0);  // removeFirst equivalent
        list.remove(list.size() - 1); // removeLast equivalent

        System.out.println("After removeFirst and removeLast: " + list);

        System.out.println("First element: " + list.get(0));
        System.out.println("Last element: " + list.get(list.size() - 1));

        System.out.println("Contains Mango? " + list.contains("Mango"));
        System.out.println("Index of Banana: " + list.indexOf("Banana"));

        list.set(1, "Kiwi");
```

```
        System.out.println("After set(1, Kiwi): " + list);

        list.remove("Apple");
        System.out.println("After removing Apple: " + list);

        System.out.println("Size: " + list.size());

        list.clear();
        System.out.println("After clear, isEmpty? " + list.isEmpty());
    }
}
```